

Remarks / Arguments

Inventorship and Common Ownership – 37 CFR 1.56

The inventors identified in the present application are Dr. Richard P. Walker and Mr. Christopher Sonnenberg. The Applicant respectfully advises that the entire subject matter of the present application including all claims that have been and are presently pending was commonly owned at all times by Thoughtslinger Corporation, which has employed both inventors at all relevant times. An assignment from both of the inventors to Thoughtslinger Corporation has been filed with the USPTO and is recorded at reel/frame number 012111/0482.

Dr. Walker was named as the inventor in the priority provisional application. Dr. Walker was the inventive contributor to the subject matter of the provisional application. The present application includes additional subject matter and Mr. Sonnenberg is an inventive contributor to such subject matter.

Substantive Objections/Rejections

The Applicant has amended claims 41, 44, 47-48, 60 and 66.

The Applicant has canceled claim 78, without prejudice.

The Examiner objected to the use of a lowercase "i" to identify a subparagraph of claim 41. In the Applicant's response to the previous office action, the lowercase "j" was used to identify this subparagraph of the claim. However, for clarity, the Applicant has shown an amendment in claim 41 to change the lowercase "i" which seems to be set out in the Examiner's copy of the claims to a lowercase "j" to identify the subparagraph. The subparagraphs now proceed from "h" to "j". Corresponding changes have been made elsewhere in claim 41 and also in claim 66.

Amended claim 41 relates to a method of simultaneous multi-user document editing by a plurality of users including a first user and a second user. The document includes primary data.

According to the method of claim 41 as amended, the primary data is divided into three or more mutually exclusive sections, including a first section and a last section.

Each of the sections is stored in a separate primary container. Each of the primary containers is a sibling container and the primary containers are part of a master document tree data structure that is stored in a file system accessible to a server. The primary container in which the first section is stored is designated a head primary container. The primary container in which the last section is stored is designated a tail primary container. The master document tree data structure also includes a parent container.

The primary containers are linked together to form a linked list in which the order of the primary containers corresponds to the order of the sections in the document. The head primary container is at the head of the linked list and the tail primary container is at the tail of the linked list.

The parent container includes a link to the head primary container and to the tail container. The parent container does not include a direct link to the other primary containers in the linked list.

A copy of at least part of the master document tree is transmitted to a first client operated by the first user. A copy of at least part of the master document tree is transmitted to a second client operated by a second user.

A first lock request is received from the first client and the first lock request identifies a first group of primary containers corresponding to a first part of the document.

If the first user is permitted to lock each of the containers in the first group of primary containers, then the first group of containers is locked and is identified as being locked by the first user. A first confirm lock message is sent to the first client.

A first update message is transmitted to the second client and indicates that each primary container in the first group of primary containers has been locked.

A first post request is received from the first client. The first post request includes one or more new primary containers. The new primary containers are inserted into the linked list of primary containers in the master document tree data structure.

A second update message is sent to the second client and includes the one or more new primary containers.

The method of claim 41 as amended allows a first user to lock a portion of a document, while a second client is informed about the locked portion. The first user subsequently posts new sections of the document. The master document tree data structure is updated in accordance with the post request and the second client is informed about the new sections of the document.

Step (k)(iv) has been amended to indicate that the first post request includes one or more new primary containers corresponding to a modified version of the first part of the document. Step (k)(v) has been amended to indicate that modification of the master document tree in accordance with the first post request includes inserting the one or more new primary containers into the linked list of primary containers. Step (k)(vi) has been amended to indicate that the second update message includes the one or more new primary containers.

In the master document tree data structure, the document sections are stored in three or more sibling containers that are arranged in a linked list. The parent container contains links to the head and tail sibling containers, but does not include a direct link to any containers in the linked list between the head and tail containers. The number and order of sibling containers is not maintained by the parent container.

The Examiner rejected claim 41 as previously on file as obvious in light of Bray and Devanbu. Bray indicates that his authoring system and his tree structure are based on XML. Referring to Bray column 1, lines 46-50, the contemporaneous XML specification is incorporated by reference into Bray. In column 5, lines 8-9, Bray states "XML is used to define the allowable relationships between entities", wherein an entity is a node in a document tree. Bray's adherence to XML is reinforced in column 5, lines 9-12 wherein he states "An XML document or fragment of a document will have a single "root" entity that is the parent of all of the underlying subtree. A single node can have many children but a node can have only one parent." XML documents follow a well-known tree structure in which the document has a root node, nodes may have many children but only one parent, and so-called 'leaf nodes' reside at the end of branches of the tree. A parent node contains a direct link to all of its child nodes. Child nodes are not linked to one another and are not linked into a linked list. Further confirmation of this structure appears in Bray Figure 3.

In XML and Bray, the number and ordering of child nodes is recorded in the parent node. To add (create) new child nodes to a parent node, or delete existing child nodes from a parent node, the parent node must be accessed. This is reflected in Bray's locking rules for creating new child nodes. In column 8, lines 4-5, Bray indicates that in order to create a new child node, a create lock is placed on the parent node. In column 8, lines 18-23, he describes how the parent node's vector of children is affected by creation of a new child node, requiring a lock on the parent node. In column 8, lines 35-36, he states that if a lock of any kind exists on the parent node, the create lock request is denied, which means that the client is unable to add a new child node. Similar restrictions are placed on deletion of child nodes. In column 8, lines 64-67, he indicates that in order to delete an existing child node, a delete lock is placed on both the parent and child nodes. In column 9, lines 20-27, he states that if a lock of any kind exists on either the child or the parent node, the delete lock request is denied, which means that the client is unable to delete a child node. In column 9, lines 58-61, Bray describes how the parent node's vector of children is affected by deletion of a child node, requiring a lock on the parent node.

In contrast, in the method of claim 41 as amended, the document tree data structure differs from and is illegal under XML and thereby under Bray. Primary containers in which adjacent sections of the document are stored are linked together in a linked list that corresponds to the order of the sections in the document. The parent container does not contain a link to primary containers between the head and tail primary containers. The number and order of primary containers is not maintained by or stored in the parent container. As a result, primary containers may be added to the linked list of primary containers without first locking the parent container.

During a multi-user document editing session, new primary containers are frequently posted and update messages are frequently generated. For example, in multi-user text document editing, new paragraphs (corresponding to new primary containers) are frequently created and added to a document. To permit simultaneous document editing, there must be as few impediments as possible.

Bray precludes simultaneous multi-user document editing by multiple users if a section of a document is added or deleted by any of the users. In Bray, clients must ensure that the parent node is unlocked before new child nodes may be created and added. If the parent node is locked, the client must make repeated create lock requests until the parent node is unlocked. If a create lock is granted, it is placed on the parent node until the client explicitly requests its removal (column 8 lines 59-60). Bray's rules for create locks, and the requirement of a separate transaction to release the create lock on the parent node, impede the addition of new child nodes, thereby rendering simultaneous multi-user document editing impossible. In the method of claim 41 as amended, locks are established on primary containers only to the extent necessary to prevent co-editing conflicts. New primary containers are added to the document without regard to the lock state of the parent container, which is simply impossible in Bray. The method of claim 41 as amended thereby enables simultaneous multi-user document editing.

Referring to Devanbu column 5, lines 1-19, Devanbu describes various kinds of parts that may be shared between users in a multi-user domain. A single user part is controlled by a single user but viewed by all users (lines 7-9). A single thread part is checked out by a user, modified by the user, then checked back in, such that the part is modified by only

one user at a time (lines 12-17). A fully concurrent part may be modified by more than one person at once (lines 17-19). Devanbu cautions against full concurrency. In column 5 lines 23-25, Devanbu states "fully concurrent parts should be used only in the most unusual circumstances, since this type of part demands a great deal of traffic over the communications medium". In column 5, lines 33-34, Devanbu further recommends that "the lowest acceptable level of concurrency should be chosen".

In column 5 lines 56-67, Devanbu describes the work that would be necessary to create a multi-user domain client, without enumerating any of the required "about 60" functions. In column 5 line 67 to column 6 line 2, he states that such a client does "not handle the distributed and concurrent aspects" of document sharing. In column 6, lines 6-36, he discusses creation of wrappers for part editors in order to achieve a level of concurrency. He gives an overview of part editor wrappers in which one user may edit a part while another user is an observer (low level of concurrency). As the first user edits the part, screen redisplay calls are intercepted, sent to the server and broadcasted to other clients. Storage calls are also intercepted and sent to the server but are not broadcasted to other clients. Devanbu fails to teach any details regarding fully concurrent part editor wrappers. In column 6, lines 37-42, Devanbu states that in order to achieve higher levels of concurrency, wrappers may not be sufficient and "it may be necessary to modify the part editors themselves". Devanbu fails to suggest or teach how existing editors are modified to achieve full concurrency.

In column 4, lines 1-7, Devanbu states that the server receives a "part modification signal" from a client, and that the signal "can be sent to other clients in accordance with a concurrency model". Devanbu fails to describe any such signal other than the screen redisplay information discussed above. Devanbu fails to suggest or teach the structure, content, frequency or other important aspects of such signals in a fully concurrent model.

Devanbu also fails to address co-editing conflicts during fully concurrent part editing. If a first user and a second user are both editing the same part at the same time (column 5, lines 17-19), a mechanism must be in place to resolve co-editing conflicts in which both users wish to edit the same section of the part at the same time. Devanbu fails to suggest or teach any method of resolving co-editing conflicts in a fully concurrent model.

Devanbu does not teach any methods for creating client and server side wrappers for, or directly modifying, existing editors (such as Microsoft Word) to achieve fully concurrent multi-user editing with resolution of co-editing conflicts. Furthermore, in the Applicant's respectful submission, such methods would not be obvious to a person having ordinary skill in the art.

In contrast, in the method of claim 41 as amended, simultaneous (fully concurrent) multi-user document editing is enabled. A document is divided into mutually exclusive sections that are stored in primary containers. Primary containers are locked to the extent necessary to prevent co-editing conflicts. Requests and update messages maintain the master document tree, inform other users of locking and editing activity, and do not demand a great deal of network traffic as is asserted by Devanbu. Existing editors are neither wrapped nor modified. The highest level of concurrency is achieved.

In summary, the teachings of Bray are inconsistent with the method of claim 41. The teachings of Devanbu are clearly directed away from the use of a fully concurrent editor and furthermore are insufficient to develop such an editor. The Applicant submits that Bray and Devanbu cannot be combined to provide the method of claim 41.

Claims 42-55 and 57-59 are dependent on claim 41. Claims 44 and 48 have been amended to be consistent with claim 41, as amended.

Claim 47 has been amended to be dependent on and consistent with claim 41.

Claim 60 is dependent on claim 41. According to the method of claim 60 as amended, editable summary information is stored in the parent container. A second lock request is received from the second client and identifies the parent container. If the second user is permitted to lock the parent container, then the parent container is locked (step (q)(i)). A confirm lock message is transmitted to the second client. A third update message is transmitted to the first client indicating that the parent container has been locked. A second post request is received from the second client and contains modified summary information. The master document tree is modified in accordance with the second post

request. A fourth update message is transmitted to the first client and includes a modified version of the parent container. The first post request (see claim 41, step (k)(iv)) includes at least one new section. Step (q)(i) occurs before (k)(iv).

According to the method of claim 60, a first user can lock a first part of the document that contains one or more sections of the document. The second user then locks the parent container and edits the information stored therein. The first user then adds at least one new section to the first part of the document.

As noted above, in Bray's tree structure, the parent node maintains the number and order of child nodes and the parent node must have a link to each of its child nodes. If a new section is added to the document, the parent node must be amended to insert the new section at the appropriate position and a link must be established to the corresponding new child node. As a result, Bray does not permit one user to add a new section to a document while another user has a lock on the parent node. See Bray at column 8, lines 1-60. Accordingly, Bray is inconsistent with the method of claim 60 for this reason in addition to the reasons set out above in relation to claim 41.

With respect to claim 60, the Examiner also made reference to Barlow. Barlow describes interactive objects for which behavior may be locked by the object designer (abstract). Referring to Barlow Figure 13, the object data structure stores an attribution pointer that points to an attribution data structure, which contains a summary of attribution information such as a copyright notice (column 20 lines 37-49). Referring to Figure 14, the attribution data structure stores a database pointer that points to more detailed attribution information. Attribution information is entered by the object designer (column 21 lines 5-7) and is locked to prevent others from modifying it (column 6, lines 5-12). Access to attribution information is password protected to prevent others from changing it (column 24 lines 1-10).

In the method of claim 60 as amended, the parent container stores editable summary information, possibly along with other information. See step 60(k)(iv) which indicates that the second post request includes a modified version of the parent container, which has been received from the second client. At least part of the information stored in the parent

container is editable by the second user. In contrast, Barlow does not teach a document tree data structure and does not permit one user to add new sections to a document while a parent container that contains editable summary information is locked and edited by another user. Accordingly, Barlow is inconsistent with the method of claim 60. The Examiner also made reference to Devanbu in respect of claim 60. However, nothing in Devanbu appears to be relevant to the method of claim 60. Bray, Devanbu and Barlow could not be combined to provide the method of claim 60.

With respect to claim 66, the Examiner additionally made reference to Madduri. Madduri describes an access control program in which a list of users along with their privileges is maintained. Madduri's system restricts access to entire data objects or documents (see column 3, lines 28-34 and column 4, lines 1-4). Accordingly, Madduri does not require or teach the use of a document tree or the division of a document into sections. Madduri does not describe storing a unique user handle for a user in a container (or a document) along with privilege levels for the user to control access to other contents of the container (or the document) (see column 4, lines 5-14). Accordingly, Bray, Devanbu and Madduri cannot be combined to provide the method of claim 66. In any case, claim 66 includes the limitations of claims 41 and is patentable for the reasons set out above.

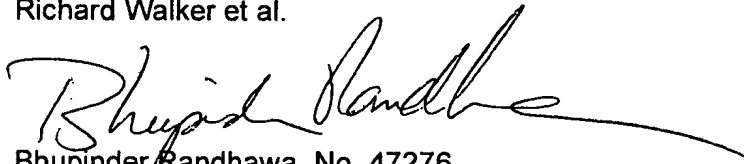
With respect to claim 75, in the Office Action, the examiner makes reference to Bray at col. 7, lines 51-67. This section of Bray addresses the retention of different versions of a document. Such different versions may be useful to a user who wishes to discard a later version of a document and return to an earlier version of it and for other purposes. Such decisions are generally made between editing sessions. This section does not relate to change tracking as identified in claim 75. Change tracking is generally intended to allow users to view the editing activities, such as insertions and deletions, of other users during and after an editing session.

Conclusion

In view of the foregoing comments, it is respectfully submitted that the application is now in condition for allowance. If the Examiner has any further concerns regarding the language of the claims or the applicability of the prior art, the Examiner is respectfully requested to contact the undersigned at 416-957-1630.

Respectfully submitted,

Richard Walker et al.

A handwritten signature in black ink, appearing to read "Bhupinder Randhawa", with a long horizontal flourish extending to the right.

Bhupinder Randhawa, No. 47276

Bereskin & Parr, Customer No. 001059